作业3报告

xxx

(南京大学 xxx, 南京 210093)

# 学习方法的尝试与介绍

### 1. 多层感知机（MLP）

**简介**  
多层感知机是一种前馈人工神经网络，适用于监督学习任务。它由输入层、隐藏层和输出层组成，其中每个隐藏层包含若干神经元，通过非线性激活函数连接。Sklearn中的MLP使用反向传播算法调整权重，使得预测值接近真实值。

**特点**

* **优势**: 非线性建模能力强，适合处理复杂数据特征。性能最好（战绩可查: [./logs/神经网络(通关版)](logs/神经网络(通关版)/replay_ai.gif)）
* **劣势**: 模型复杂度高，对参数较敏感，训练速度较慢。Aliens测试中在包括RandomForest的四个模型中训练时间最长（4mins）。

**性能**  
在此任务中，MLP的表现最佳，表明它可以有效捕捉提取的网格特征和动作之间的复杂关系。

### 2. 决策树（Decision Tree）

**简介**  
决策树是一种基于树结构的分类模型。每个节点表示一个特征上的决策规则，叶子节点对应预测的类别。决策树通过递归地选择最优分割点，最大化信息增益或减少不纯度。

**特点**

* **优势**: 易于理解和可解释性强，能快速训练和预测。([./logs/决策树(几乎通关版)](logs/决策树2（几乎通关版）/replay_ai.gif))
* **劣势**: 容易过拟合，尤其在特征数量多或数据样本少的情况下。这个情况在只用两个训练样本训练level0的时候比较突出。上面的测试gif图中很容易看出，由于训练样本的不稳定性，导致该模型在模仿和探索之间取舍失衡，最后陷入到周期性的运动中。

**性能**  
决策树在此任务中的表现优于KNN，但逊色于MLP。尽管其决策规则简单有效，但对于复杂特征之间的关系建模能力不足。同时在样本数量较少的时候，容易出现过拟合的现象.

### 3. K近邻（KNN）

**简介**  
K近邻是一种基于实例的非参数模型。预测时，它计算测试样本与训练样本之间的距离，并选择距离最近的K个样本进行投票或加权平均，输出类别或回归值。

**特点**

* **优势**: 实现简单，训练速度快。
* **劣势**: 对特征空间的分布敏感，内存需求高，计算复杂度随着数据规模增大而显著增加。尤其Alien项目中的数据量是（1k量级的步骤 \*（32\*15）方格 \* 9个格子状态），维度和数量都比较高，这使得KNN的计算任务相当巨大，性能比较低下。

**性能**  
KNN在此任务中的表现最差。主要原因可能是提取的特征维度较高，而KNN在高维空间中的分类能力受到“维度灾难”的影响，导致分类效果下降。

### 4. 随机森林（Random Forest）

**简介**  
随机森林是一种基于决策树的集成学习方法，通过训练多个决策树并将它们的结果组合（如投票或加权平均）来提升模型的准确性和稳定性。它利用随机子采样和特征选择，降低过拟合的风险，同时提高了模型的泛化能力。

**特点**

* **优势**:
  + 通过集成多个决策树，有效减少单棵树的过拟合问题。
  + 在处理高维数据和复杂特征时表现良好。
* **劣势**:
  + 训练过程需要更多计算资源，尤其是当树的数量较多时。
  + 参数调优（如树的数量、最大深度等）较复杂。

**性能**  
在当前任务中，随机森林性能优于KNN但稍逊于单棵决策树。这可能是因为随机森林的集成特性在减少过拟合时牺牲了一定的决策灵活性，而提取的特征与目标动作之间的关系较适合直接使用单棵决策树。

### 4. 性能对比

| **学习方法** | **表现排名** | **优势** | **劣势** |
| --- | --- | --- | --- |
| **MLP** | 1 | 强大的非线性建模能力 | 需要较多计算资源，参数调优复杂 |
| **决策树** | 2 | 可解释性强，训练速度快 | 容易过拟合，复杂特征建模能力有限 |
| **随机森林**  **KNN** | 3  4 | 减少过拟合，能处理高维特征和非线性关系  实现简单，无需训练 | 训练时间较长，模型复杂度较高  对高维数据表现不佳，预测时间长 |

代码见learn.py和test.py中#所标注部分:

1. ##################################################
2. **from** sklearn.ensemble **import** RandomForestClassifier
3. **from** sklearn.neighbors **import** KNeighborsClassifier
4. **from** sklearn.tree **import** DecisionTreeClassifier
5. **from** sklearn.neural\_network **import** MLPClassifier
6. ##################################################
8. ##############################################
9. clf = RandomForestClassifier(n\_estimators=100)
10. clf = KNeighborsClassifier(n\_neighbors=3)
11. clf = DecisionTreeClassifier()
12. clf = MLPClassifier(hidden\_layer\_sizes=(100, 50, 25), max\_iter=1000)
13. ##############################################

# 特征提取方法的修改

仿照原代码通过提取每个单元格的物体信息生成特征向量的思路，我对 extract\_features 函数进行了改进，新增了两种特征提取方法，旨在提升模型对全局布局和动态场景的理解能力。具体修改如下：

### 相对位置的判断

**方法描述：**  
在原有特征的基础上，增加了基于“玩家”（avatar）位置的相对位置特征。

* 计算“外星人”（alien）和“炸弹”（bomb）与“玩家”之间的横向距离（曼哈顿距离的横向分量）。
* 若与“avatar”存在相对位置关系，则取第二小的距离（防止取到0）生成一个对应的距离特征。

**优势**：

* 增强了特征的动态性，特别是在需要基于“玩家”位置规划动作的任务中。
* 能够更好地捕捉与“玩家”相关的关键威胁或目标物体的分布。

**实现代码关键实现逻辑：**

1. avatar\_position = None
2. col = len(grid[0])
4. **for** row **in** grid:
5. **for** j, cell **in** enumerate(row):
6. **if** 'avatar' **in** cell:
7. avatar\_position = j
8. **break**
9. **if** avatar\_position **is** **not** None:
10. **break**
12. **def** second\_min(distances):
13. # 如果元素不足两个，返回默认值
14. **if** len(distances) < 2:
15. **return** 1
16. # 对列表进行排序并返回第二小的值
17. unique\_distances = sorted(set(distances))  # 去重后排序
18. **return** unique\_distances[1] **if** len(unique\_distances) > 1 **else** 0
20. **def** get\_relative\_position():
21. alien\_distances = []
22. bomb\_distances = []
23. alien\_feature\_vector = [0] \* len(object\_mapping)
24. bomb\_feature\_vector = [0] \* len(object\_mapping)
25. **if** avatar\_position **is** **not** None:
26. **for** row **in** grid:
27. **for** j, cell **in** enumerate(row):
28. **for** obj **in** cell:
29. **if** obj == 'alien':
30. alien\_distances.append(abs(avatar\_position - j) / col)  # 使用曼哈顿距离的横向分量计算相对距离, 并且进行归一化
31. **if** obj == 'bomb':
32. bomb\_distances.append(abs(avatar\_position - j) / col)  # 使用曼哈顿距离的横向分量计算相对距离, 并且进行归一化
33. alien\_distance = second\_min(alien\_distances)
34. alien\_feature\_vector[3] = alien\_distance
35. bomb\_distance = second\_min(bomb\_distances)
36. bomb\_feature\_vector[4] = bomb\_distance
37. **return** alien\_feature\_vector, bomb\_feature\_vector

40. alien\_feature\_vector, bomb\_feature\_vector = get\_relative\_position()
41. **for** \_ **in** range(col):
42. features.extend(alien\_feature\_vector)
43. **for** \_ **in** range(col):
44. features.extend(bomb\_feature\_vector)

### 区域特征提取

**方法描述：**  
将网格划分为固定的4×4区域，对每个区域内的物体类别进行计数，并将这些计数作为新的特征。

对每个区域，统计“wall”、“bomb”等物体的数量。

将区域特征拼接到全局特征向量中，形成最终的输入特征。

**优势：**

提供了网格的全局分布信息，增强模型对场景布局的理解能力。

有助于捕捉区域内的局部模式，例如某一区域是否密集分布威胁物体。

**实现代码关键实现逻辑：**

1. # 提取区域特征, 但是添加以后极慢无比！
2. num\_regions = 4
3. region\_size = len(grid) // num\_regions
4. **for** region\_i **in** range(num\_regions):
5. **for** region\_j **in** range(num\_regions):
6. region\_counts = [0] \* len(object\_mapping)
7. **for** i **in** range(region\_i \* region\_size, (region\_i + 1) \* region\_size):
8. **for** j **in** range(region\_j \* region\_size, (region\_j + 1) \* region\_size):
9. **for** obj **in** grid[i][j]:
10. index = object\_mapping.get(obj, -1)
11. **if** index >= 0:
12. region\_counts[index] += 1
13. features.extend(region\_counts)

### 3. 性能对比 以下是改进特征提取方法前后的平均存活步数（测试5局）比较，**初步判断是复杂的模型在改进之后因为信息的增多而性能有所提升，但是简单的模型在信息量增大的同时导致计算任务增大和过拟合现象影响的突出导致性能反而下降。**

| 学习方法 | 原始平均存活步数 | 改进后平均存活步数 | 提升幅度 |
| --- | --- | --- | --- |
| **MLP** | 892.3 | 1022.4 | +14.6% |
| **决策树** | 504.7 | 522.8 | +3.5% |
| **随机森林** | 289.2 | 257.2 | -11.1% |
| **KNN** | 189.5 | 170.9 | -9.8% |

References:

[1] [机器学习好伙伴之scikit-learn的使用——常用模型及其方法\_nfstream机器学习可用的模型-CSDN博客](https://blog.csdn.net/weixin_44791964/article/details/100561879)

[2] [机器学习入门研究（五）-特征工程之特征提取\_机器学习特征提取的数学方法-CSDN博客](https://blog.csdn.net/nihaomabmt/article/details/102931744)